5 Testing

The checklist is dynamic, not static, and will be updated regularly. If you have any suggestions or comments, we would like to hear from you.

Test that the requirements for data protection and security that were specified in Requirements have in fact been implemented, and that they are correctly implemented:

- Remember that the data protection regulation also apply to development and testing environments.
- Establish a comprehensive understanding of functionality and information.
- Verify that the requirements and design components have fulfilled the security and data protection requirements. This can, for example, be done by creating standard test scenarios based on functional requirements that can be reused throughout the business.
- Compile a checklist on whether all the components needed for compliance are included. This also includes new components that may not originally have been specified when the requirements were determined. Examples:
    - All settings should be set to the most privacy-friendly option by default.
    - It must be possible to export and import the data subject's data (data portability).
    - Is data being saved in the correct place?
    - Is the data being collected necessary for the purpose of the software?
    - The data subject should be able to give consent (this applies also to children and persons subject to guardians).
    - The data subject must be able to refuse or withdraw consent.
    - Is it possible to terminate a contract/agreement, install, uninstall, activate and deactivate a program, service, technical component, or system?
    - Access control
        - for authorised parties only
        - accessible to the data subject
        - gives access to rectify, block, or delete personal data
    - It should be possible to send queries or complaints relating to data protection and security.
    - The data subject can reserve the right not to be profiled.
    - Get information about how automated decisions are made, what is collected and processed, where personal data is stored, transparency, etc.
- Test that notification procedures exist and work, that notifications are acknowledged (for example, develop a draft text that can be used in the event of an incident, including measures to prevent a similar incident from occurring again), as well as what steps have been taken to prevent any impact on the data subject.
- By testing use synthetic personal data to check correctness, lawfulness, fairness, purpose, accuracy, data minimisation, and resilience.
    - Use real data only for quality assurance when the software is implemented in the production environment, and users will be using the software. Tests on real data should be performed by professionals who are authorised to view the data, such as appropriate professionals for child welfare programs in the child welfare

sector, super users for patient journals, or super users for a school system in a school.

- Search for real personal data, and check why they are appearing. This can, for example, be done by regularly scanning for potential leaks of national identity numbers.

Security testing by challenging vulnerabilities in the software

- **Dynamic testing:** test the functionality of the running code by using tools or manual review to analyse how the software behaves with different user privileges and for critical security vulnerabilities. Testing must ensure that users only gain access to the information and functions they are authorised for. Verify that unauthorised attempts to acquire information are logged as security and personal data breaches.
    - Prefer input validation with whitelisting over blacklisting
    - Perform a vulnerability analysis on the application and network layers. The analysis should measure the effect of implemented technical and organisational security measures. Use testing tools. For example, test for default passwords, passwords stored in files, SQL injection, script injection, cross-site scripting, missing validation, unauthorised access to logs, backups, and temporary locations.
    - Test access control and actual access to users:
        - Example of a GET request: log in user 1, copy all links, log out user 1. Log in user 2, etc. Check all links from one user that are accessible for unauthorised users or other users. Test all levels of permissions, using at least two users at each access level.
        - Simple testing by non-technicians. Use a browser, for example, and check if GET requests in the browser give access to private URLs.
        - Advanced testing of technicians. Use burp suit or burp repeater, for example, to investigate POST requests and session cookies.
        - Parameters that can be iterated (e.g. ID=42, 43, 44, etc.) should be iterated and checked for access control.
        - Investigate cookie and session management, such as how login and access control work based on cookies. (If a new cookie is generated after logging in, is the session destroyed on the server side or solely deleted from the user's browser?)
- **Fuzz testing:** Fuzz testing is conducted by intentionally triggering errors in the software. This can be done using tools that send random and malformed data (incorrect input values) to all possible input fields in the software, either manually or using intelligent tools that analyse vulnerabilities in web applications. If the software has multiple interfaces, efforts should be made to test each of them. Use tools that can analyse output and apply security considerations.
- **Penetration testing/vulnerability analysis:** Perform penetration tests or vulnerability analysis on newly developed software before release, or at regular intervals, to uncover vulnerabilities. Ideally, different testers should be used every time. Security tests, such as this, must be legal and authorised attempts to find, exploit, and detect vulnerabilities. Following this, measures to make the software more robust should be implemented. Note that it may be necessary to have signed agreements in place for penetration testing.

    - This can be done both internally and externally. Assess what is most appropriate in terms of new functionality/services.
    - Use security experts to both perform testing and analyse the results.
    - Test the security features of data protection by design. Examples of vulnerabilities that may be detected are passwords, installation of Remote Controls in web browsers, clients, or servers, database dumping, data transfer, or stored data including caching, interrupting, or session and connection hijacking, exploitation of cookies, encryption cracking, exploitation of inadequate access control, etc.
- Test in multiple instances
    - Test environments, such as code review of the application (testing for vulnerabilities in the code).
    - Integration and system testing environments, such as static and dynamic tests of applications.
    - Testing in production instances, such as application fuzzing, scanning for vulnerabilities, and penetration testing of applications and infrastructure.
- Implement a regime for automatic execution of test sets before release, such as
    - security tests that are run every time an application is built or further developed. If errors occur during production,
    - similar errors can be avoided by creating an automated test for the error, and adding it to the test set.

Examples of tools that can be used for security testing:

- OWASP testing guide to OWASP's top 10 vulnerabilities, SANS/CIS 20 vulnerabilities
- White box fuzz testing
- Burp suite and burp repeater for code review
- Bug bash and "Feedback Hub".

Review the attack surface

- Verify that attack vectors discovered in the design phase have been addressed.
- Verify that new attack vectors introduced during implementation are identified and addressed.
- Review the threat model in relation to newly developed software.
- Re-assess data protection impacts that were assessed during the requirements phase, as requirements can change during the development process without being assessed.
- Pay particular attention to whether more data is being transferred than what the software requires, and if such data is lacking protection or legal basis.

Why set requirements for verification and testing?

- The data protection and security requirements that were included in the test set must be implemented, and implemented correctly, cf. Articles 5, 7, 25, 35, and 11-23.
- Incorrect input values must not contribute to breaches of confidentiality, integrity, or availability, cf. Article 32.
- The attack surface must not contain vulnerabilities that can contribute to breaches of confidentiality, integrity, or availability, cf. Article 32.